LINEAR REGRESSION PROJECT

This project is about the contract work with an Ecommerce company based in New York City that sells clothing online but they also have in-store style and clothing advice sessions. Customers come in to the store, have sessions/meetings with a personal stylist, then they can go home and order either on a mobile app or website for the clothes they want.

The company is trying to decide whether to focus their efforts on their mobile app experience or their website. My aim is to help them figure it out!

Import pandas, numpy, matplotlib, and seaborn

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

I will work with the Ecommerce Customers csv file from the company. It has Customer info, suchas Email, Address, and their color Avatar. Then it also has numerical value columns:

Avg. Session Length: Average session of in-store style advice sessions. Time on App: Average time spent on App in minutes Time on Website: Average time spent on Website in minutes

In [2]:
```
customers = pd.read_csv("Ecommerce Customers")
```

Check the head of customers, and check out its info() and describe() methods.

In [3]:
```
customers.head()
```

Out[3]:

	Email	Address	Avatar	Avg. Session Length	Time on App
0	mstephenson@fernandez.com	835 Frank Tunnel\nWrightmouth, MI 82180-9605	Violet	34.497268	12.655651
1	hduke@hotmail.com	4547 Archer Common\nDiazchester, CA 06566-8576	DarkGreen	31.926272	11.109461
2	pallen@yahoo.com	24645 Valerie Unions Suite 582\nCobbborough, D...	Bisque	33.000915	11.330278
3	riverarebecca@gmail.com	1414 David Throughway\nPort Jason, OH 22070-1220	SaddleBrown	34.305557	13.717514
4	mstephens@davidson-herman.com	14023 Rodriguez Passage\nPort Jacobville, PR 3...	MediumAquaMarine	33.330673	12.795189

In [4]:
```
customers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 8 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Email                 500 non-null    object 
 1   Address               500 non-null    object 
 2   Avatar                500 non-null    object 
 3   Avg. Session Length   500 non-null    float64
 4   Time on App           500 non-null    float64
 5   Time on Website       500 non-null    float64
 6   Length of Membership  500 non-null    float64
 7   Yearly Amount Spent   500 non-null    float64
dtypes: float64(5), object(3)
memory usage: 31.4+ KB
```

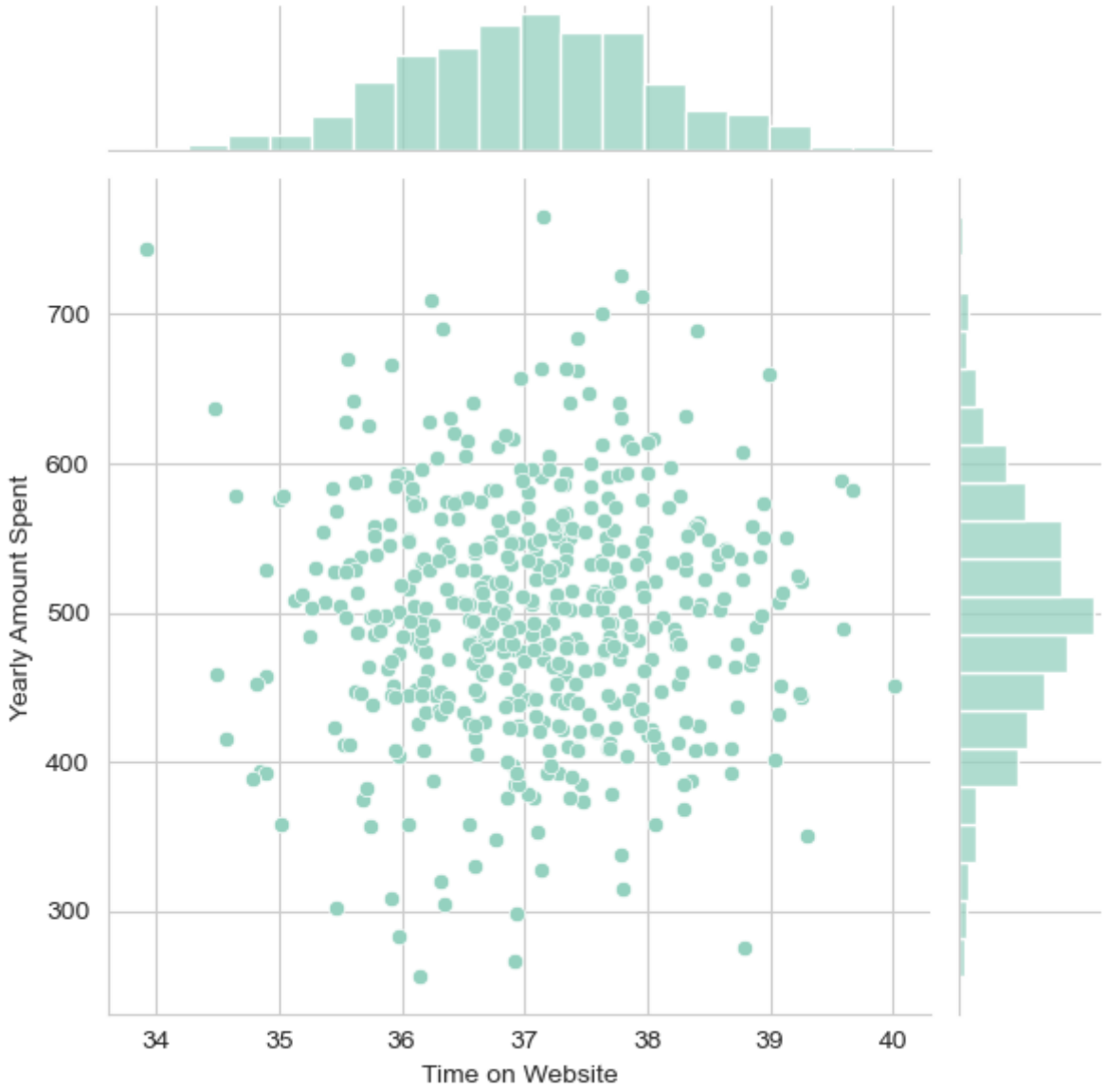
EXPLORATORY DATA ANALYSIS

For the rest of the exercise I wll only be using the numerical data of the csv file.

In [5]:
```
sns.set_palette("GnBu_d")
sns.set_style('whitegrid')
```

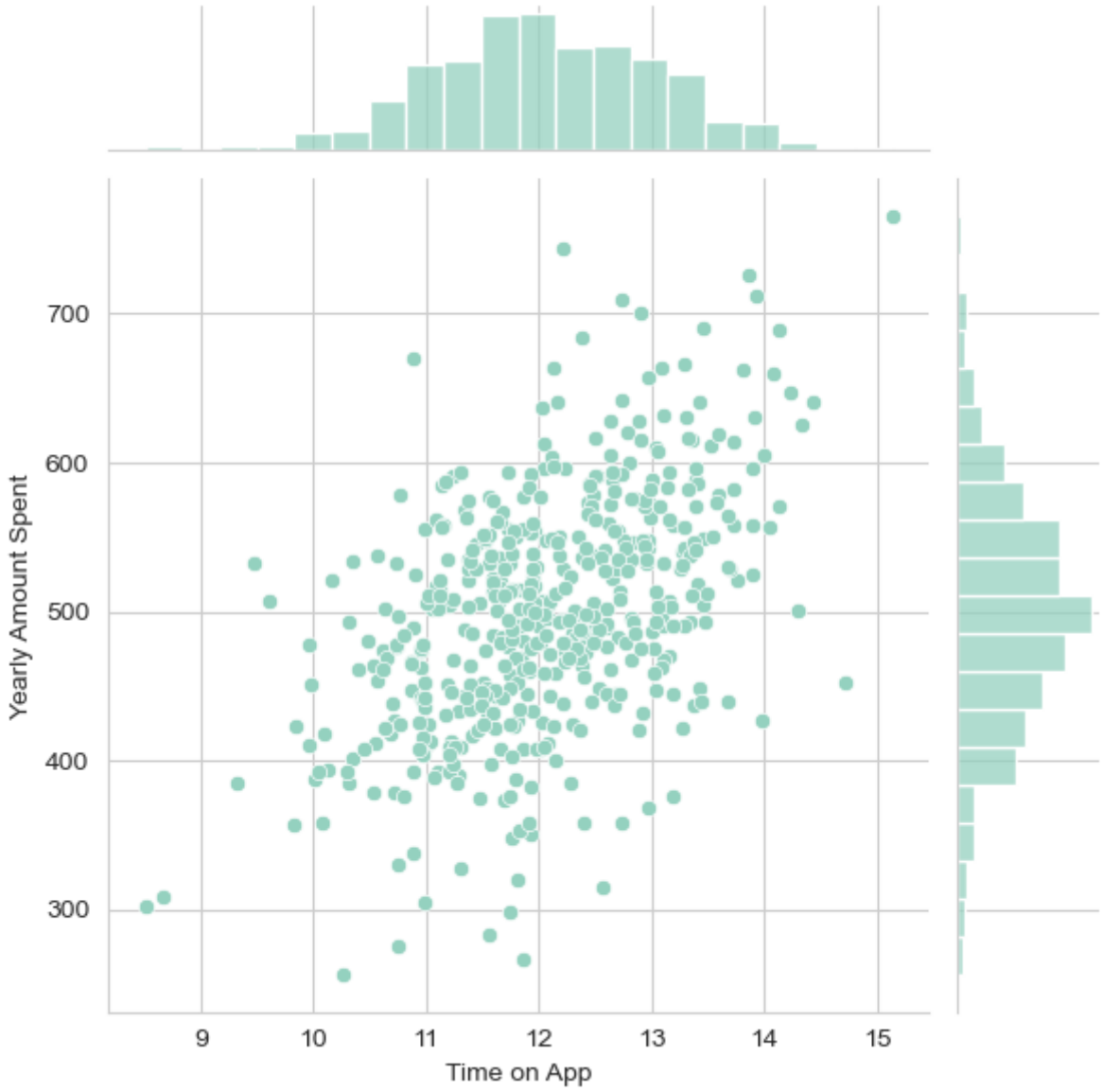
In [6]:
```
# More time on site, more money spent.
sns.jointplot(x='Time on Website',y='Yearly Amount Spent',data=customers)
```

Out[6]:
```
<seaborn.axisgrid.JointGrid at 0x12a7710c0>
```



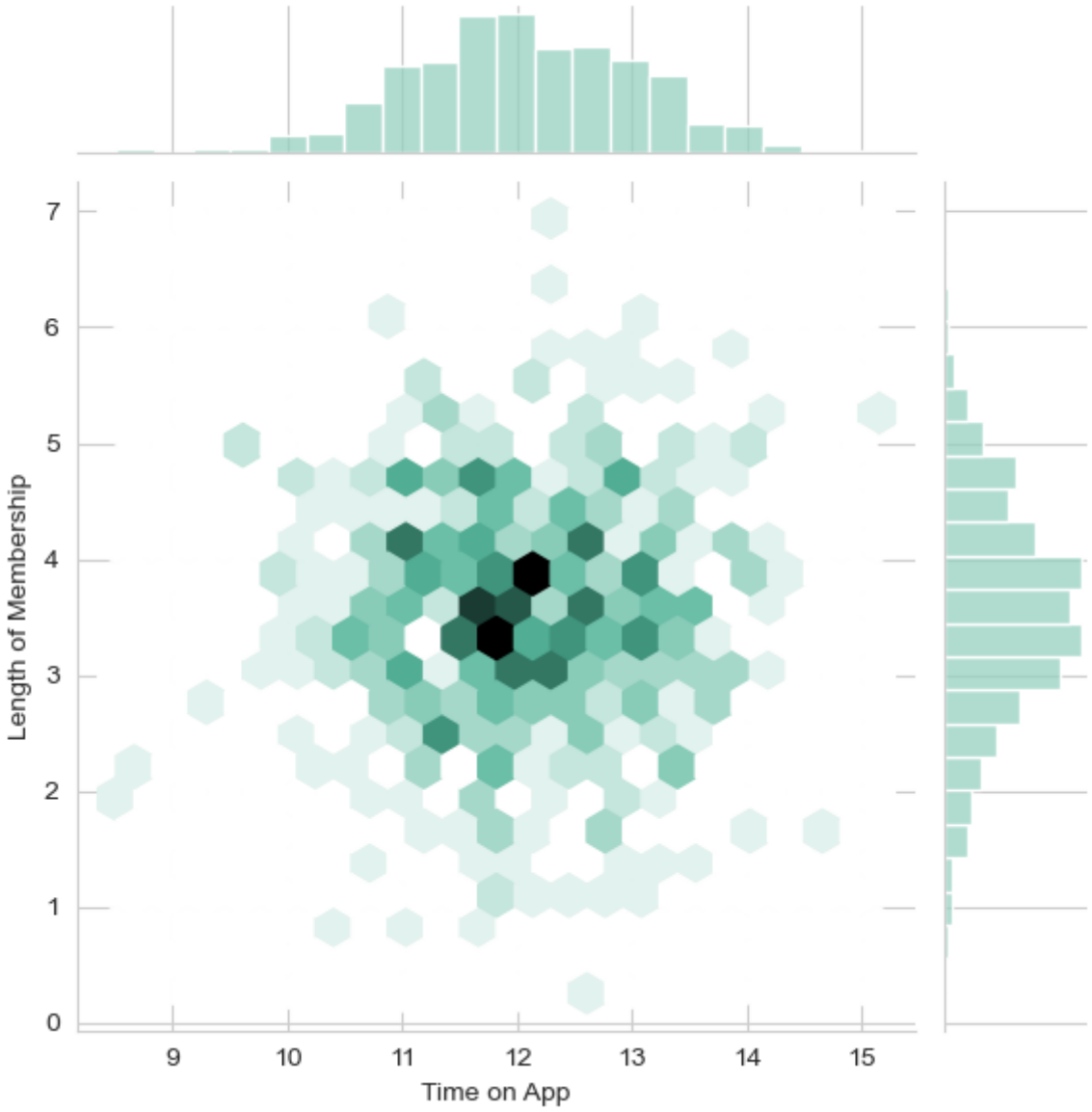
In [7]:
```
sns.jointplot(x='Time on App',y='Yearly Amount Spent',data=customers)
```

Out[7]:
```
<seaborn.axisgrid.JointGrid at 0x12a57f820>
```



In [8]:
```
sns.jointplot(x='Time on App',y='Length of Membership',kind='hex',data=customer
```

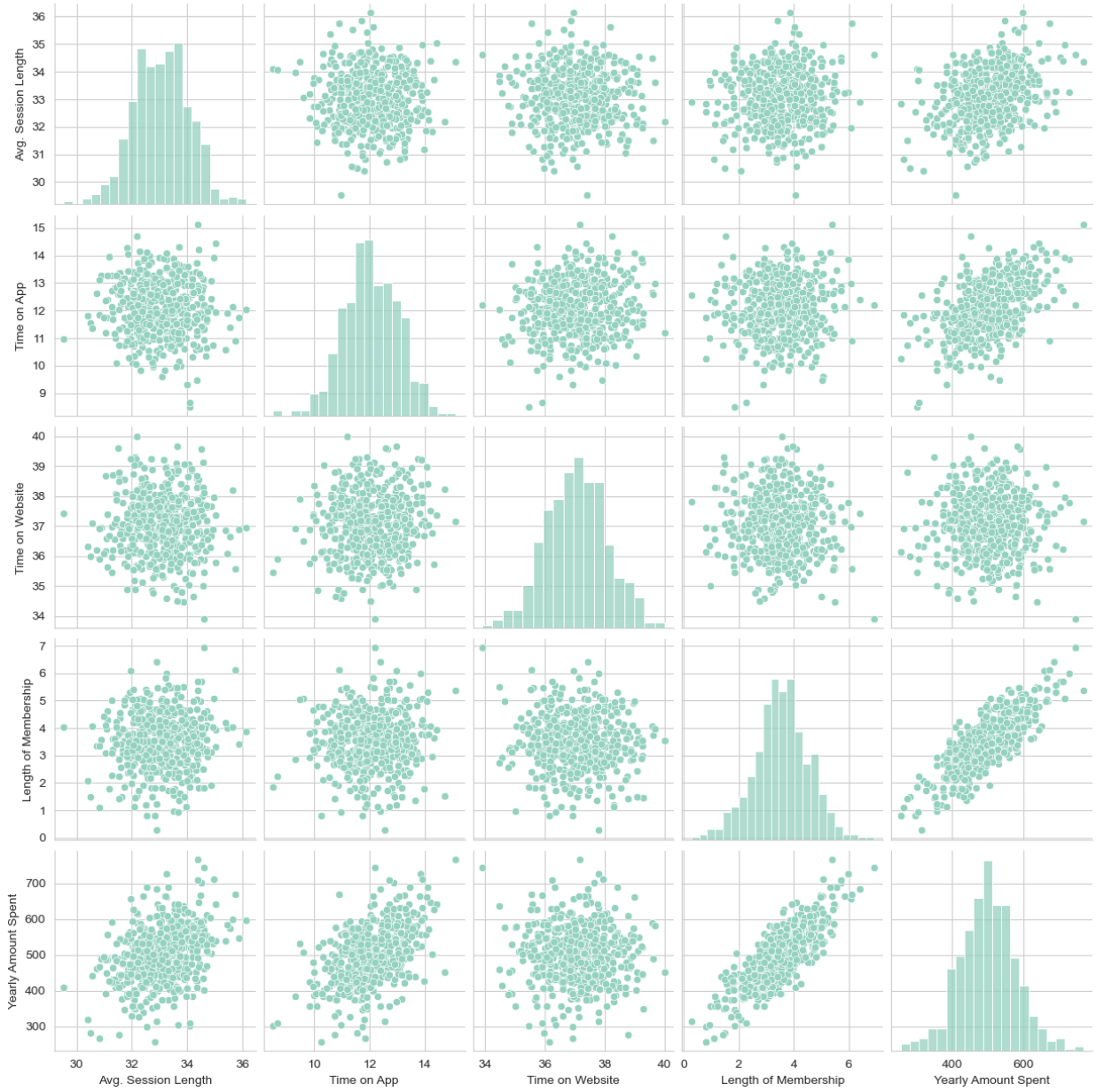
Out[8]:
```
<seaborn.axisgrid.JointGrid at 0x12ab7eb00>
```



Let's explore these types of relationships across the entire data set.

In [9]:
```
sns.pairplot(customers)
```

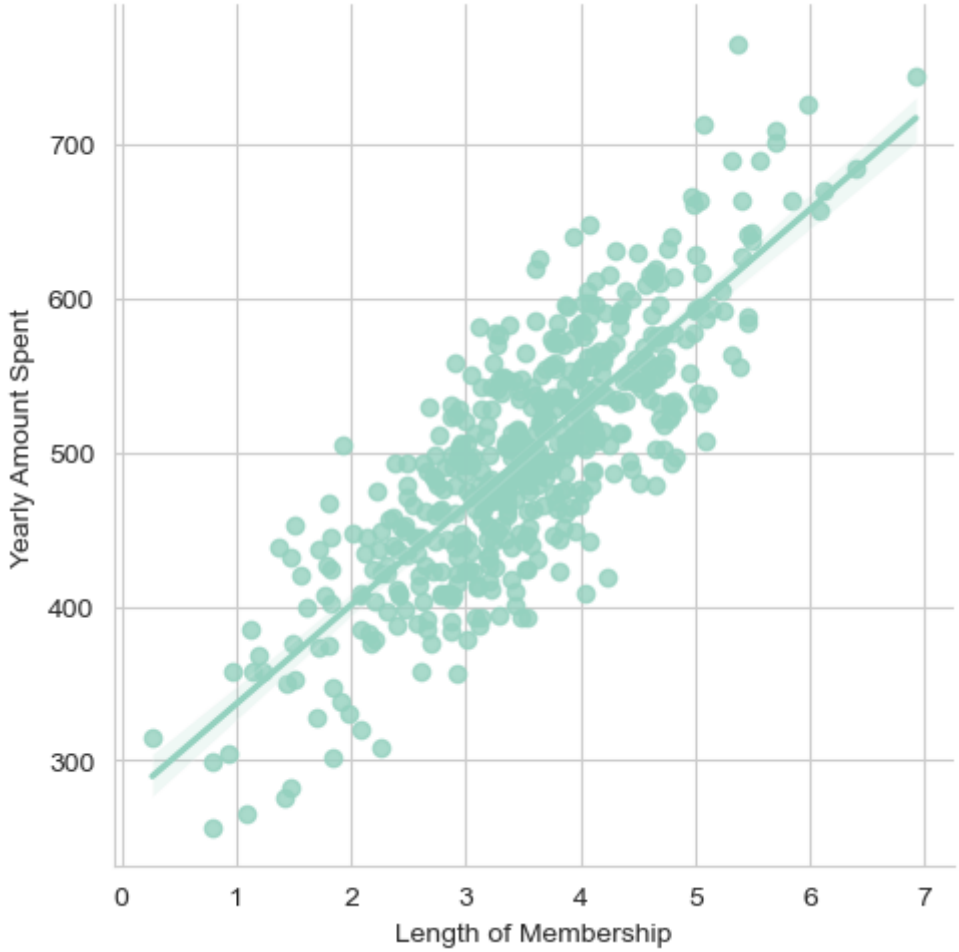
Out[9]:
```
<seaborn.axisgrid.PairGrid at 0x12ad02140>
```



I will create a linear model plot (using seaborn's lmplot) of Yearly Amount Spent vs. Length of Membership.

In [10]:
```
sns.lmplot(x='Length of Membership',y='Yearly Amount Spent',data=customers)
```

Out[10]:
```
<seaborn.axisgrid.FacetGrid at 0x12a770f70>
```



TRAINING AND TESTING DATA Now that I have explored the data a bit, I will go ahead and split the data into training and testing sets. I am going to set a variable X equal to the numerical features of the customers and a variable y equal to the "Yearly Amount Spent" column.

In [13]:
```
X = customers[['Avg. Session Length', 'Time on App','Time on Website', 'Length
```

In [14]:
```
y = customers['Yearly Amount Spent']
```

I will use model_selection.train_test_split from sklearn to split the data into training and testing sets. Set test_size=0.3 and random_state=101

In [15]:
```
from sklearn.model_selection import train_test_split
```

In [16]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random
```

TRAINING THE MODEL

Now its time to train our model on our training data!

Import LinearRegression from sklearn.linear_model

In [17]:
```
from sklearn.linear_model import LinearRegression
```

Create an instance of a LinearRegression() model named lm.

In [18]:
```
lm = LinearRegression()
```

Train/fit lm on the training data

In [19]:
```
lm.fit(X_train,y_train)
```

Out[19]:
```
LinearRegression()
```

In [20]:
```
print('Coefficients: \n', lm.coef_)
```

```
Coefficients:
 [25.98154972 38.59015875  0.19040528 61.27909654]
```

Predicting Test Data

Now that we have fit our model, let's evaluate its performance by predicting off the test values!

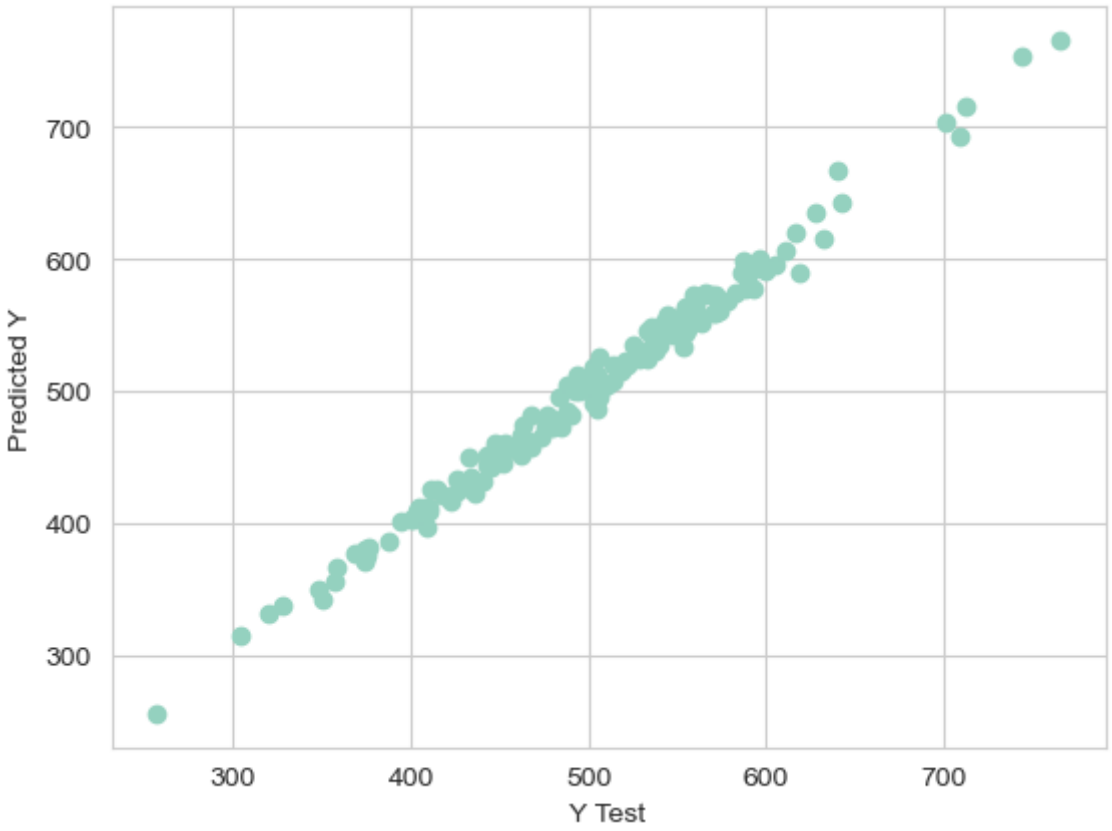
Use lm.predict() to predict off the X_test set of the data.

In [21]:
```
predictions = lm.predict( X_test)
```

Create a scatterplot of the real test values versus the predicted values.

In [22]:
```
plt.scatter(y_test,predictions)
plt.xlabel('Y Test')
plt.ylabel('Predicted Y')
```

Out[22]:
```
Text(0, 0.5, 'Predicted Y')
```



EVALUATING THE MODEL

Let's evaluate our model performance by calculating the residual sum of squares and the explained variance score (R^2).

I will then calculate the Mean Absolute Error, Mean Squared Error, and the Root Mean Squared Error.

In [23]:
```
from sklearn import metrics

print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
```

```
MAE: 7.228148653430815
MSE: 79.81305165097409
RMSE: 8.933815066978614
```

Residuals

I will plot a histogram of the residuals and make sure it looks normally distributed.

In [24]:
```
sns.distplot((y_test-predictions),bins=50);
```

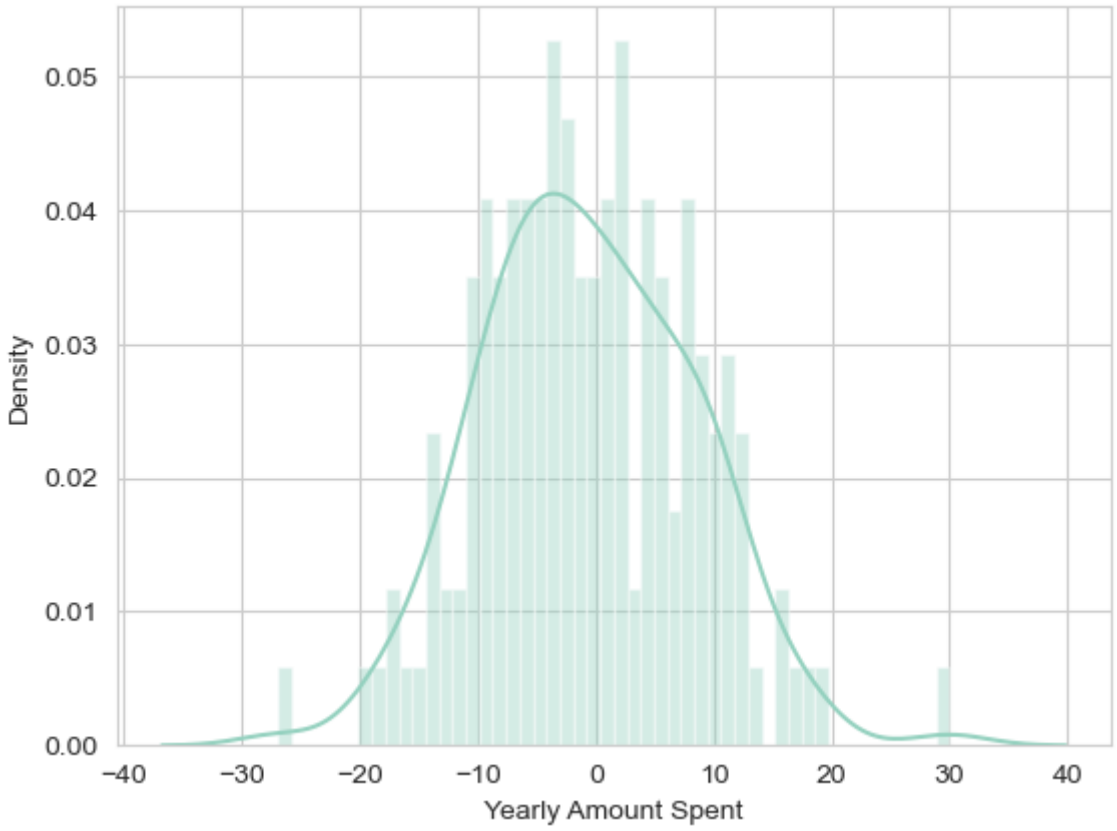
```
/var/folders/15/bdksz9nj30gfrtmt__dlzxdh0000gn/T/ipykernel_46125/1326397652.p
y:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot((y_test-predictions),bins=50);
```



Conclusion

I can check with the coefficients as well

In [25]:
```
coeffecients = pd.DataFrame(lm.coef_,X.columns)
coeffecients.columns = ['Coeffecient']
coeffecients
```

Out[25]:

	Coeffecient
Avg. Session Length	25.981550
Time on App	38.590159
Time on Website	0.190405
Length of Membership	61.279097